Lending Club Case Study
EDA assignment

Data understanding

- Variables with NA details are available here
- Loading the dataset into jupyter notebook

- Identified missing data with NA values

```
#importing required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

```
#pandas set_option allows us to see all the columns
pd.set_option("display.max_columns",None)
```

```
#reading dataset into data frame
dataset=pd.read_excel(r"E:\Data Science\Case Studies\For Resume\Project 2\loan.xlsx")
#printing first five rows of dataset
dataset.head()
```

```
#=Checking what are the features with NaN values
features_with_na=[feature for feature in data.columns if data[feature].isna().sum()>0]
features_with_na
```

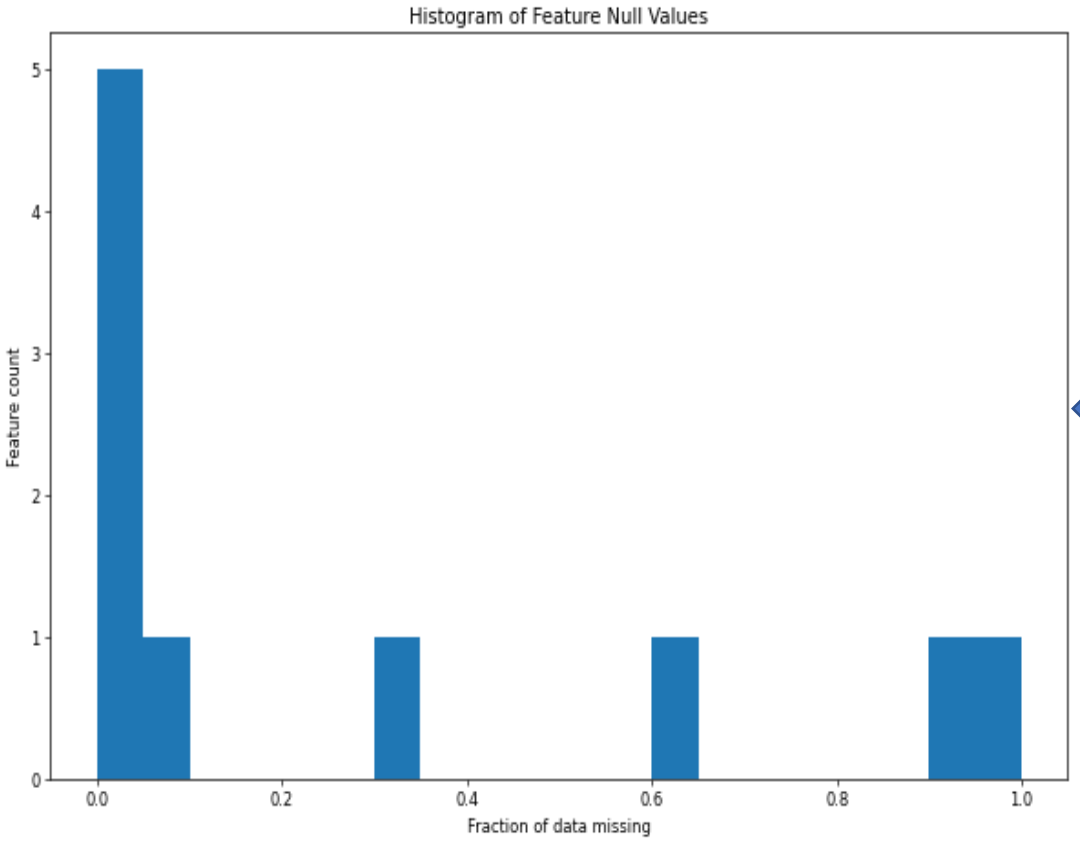
```
['emp_title',
 'emp_length',
 'desc',
 'title',
 'mths_since_last_delinq',
 'mths_since_last_record',
 'revol_util',
 'last_pymnt_d',
 'next_pymnt_d',
 'last_credit_pull_d']
```

#	Column	Non-Null Count	Dtype
0	id	39717 non-null	int64
1	member_id	39717 non-null	int64
2	loan_amnt	39717 non-null	int64
3	funded_amnt	39717 non-null	int64
4	funded_amnt_inv	39717 non-null	float64
5	term	39717 non-null	object
6	int_rate	39717 non-null	float64
7	installment	39717 non-null	float64
8	grade	39717 non-null	object
9	sub_grade	39717 non-null	object
10	emp_title	37258 non-null	object
11	emp_length	38642 non-null	object
12	home_ownership	39717 non-null	object
13	annual_inc	39717 non-null	float64
14	verification_status	39717 non-null	object
15	issue_d	39717 non-null	datetime64[n
16	loan_status	39717 non-null	object
17	pymnt_plan	39717 non-null	object
18	url	39717 non-null	object
19	desc	26777 non-null	object
20	purpose	39717 non-null	object
21	title	39705 non-null	object
22	zip_code	39717 non-null	object
23	addr_state	39717 non-null	object
24	dti	39717 non-null	float64
25	delinq_2yrs	39717 non-null	int64
26	earliest_cr_line	39717 non-null	datetime64[n
27	inq_last_6mths	39717 non-null	int64
28	mths_since_last_delinq	14035 non-null	float64
29	mths_since_last_record	2786 non-null	float64

28	mths_since_last_delinq	14035 non-null	float64
29	mths_since_last_record	2786 non-null	float64
30	open_acc	39717 non-null	int64
31	pub_rec	39717 non-null	int64
32	revol_bal	39717 non-null	int64
33	revol_util	39667 non-null	float64
34	total_acc	39717 non-null	int64
35	initial_list_status	39717 non-null	object
36	out_prncp	39717 non-null	float64
37	out_prncp_inv	39717 non-null	float64
38	total_pymnt	39717 non-null	float64
39	total_pymnt_inv	39717 non-null	float64
40	total_rec_prncp	39717 non-null	float64
41	total_rec_int	39717 non-null	float64
42	total_rec_late_fee	39717 non-null	float64
43	recoveries	39717 non-null	float64
44	collection_recovery_fee	39717 non-null	float64
45	last_pymnt_d	39646 non-null	datetime64[ns]
46	last_pymnt_amnt	39717 non-null	float64
47	next_pymnt_d	1140 non-null	datetime64[ns]
48	last_credit_pull_d	39715 non-null	datetime64[ns]

Data manipulation,

Highest Missing values, list of variables available next next_pymnt_d 1.000000 mths_since_last_record 0.928973 mths_since_last_delinq 0.645592 desc 0.324727 emp_title 0.061850 emp_length 0.026778 last_pymnt_d 0.001840 revol_util 0.001296 title 0.000311 last_credit_pull_d 0.000052




```
#Checking fractional amount of data missing in the features with null values
missing = data[features_with_na].isnull().mean().sort_values(ascending=False)
missing
```

```
next_pymnt_d              1.000000
mths_since_last_record    0.928973
mths_since_last_delinq    0.645592
desc                      0.324727
emp_title                 0.061850
emp_length                0.026778
last_pymnt_d              0.001840
revol_util                0.001296
title                     0.000311
last_credit_pull_d        0.000052
dtype: float64
```

```
[9]: #Plotting histogram for this fractional data
     plt.figure(figsize=(12,8), dpi=60)
     missing.plot.hist(bins=20)
     plt.title('Histogram of Feature Null Values')
     plt.xlabel('Fraction of data missing')
     plt.ylabel('Feature count')

t[9]: Text(0, 0.5, 'Feature count')
```

The presentation has a clear structure, is not too long, and explains the most important results concisely in simple language.

The recommendations to solve the problems are realistic, actionable and coherent with the analysis

```
educational             0.172308
renewable_energy        0.180000
small_business          0.265945
Name: loan_status, dtype: float64
```

```
data['title'].describe()
```

```
count                   37691
unique                  19025
top       Debt Consolidation
freq                     2015
Name: title, dtype: object
```

```
#drop the title variable as there are too many unique values present init and as it is not hlpful for our analysis
data.drop('title', axis=1, inplace=True)
```

```
data.shape
```

```
(37703, 22)
```

```
#check the number of unique values present in zip_code variable
data['zip_code'].nunique()
```

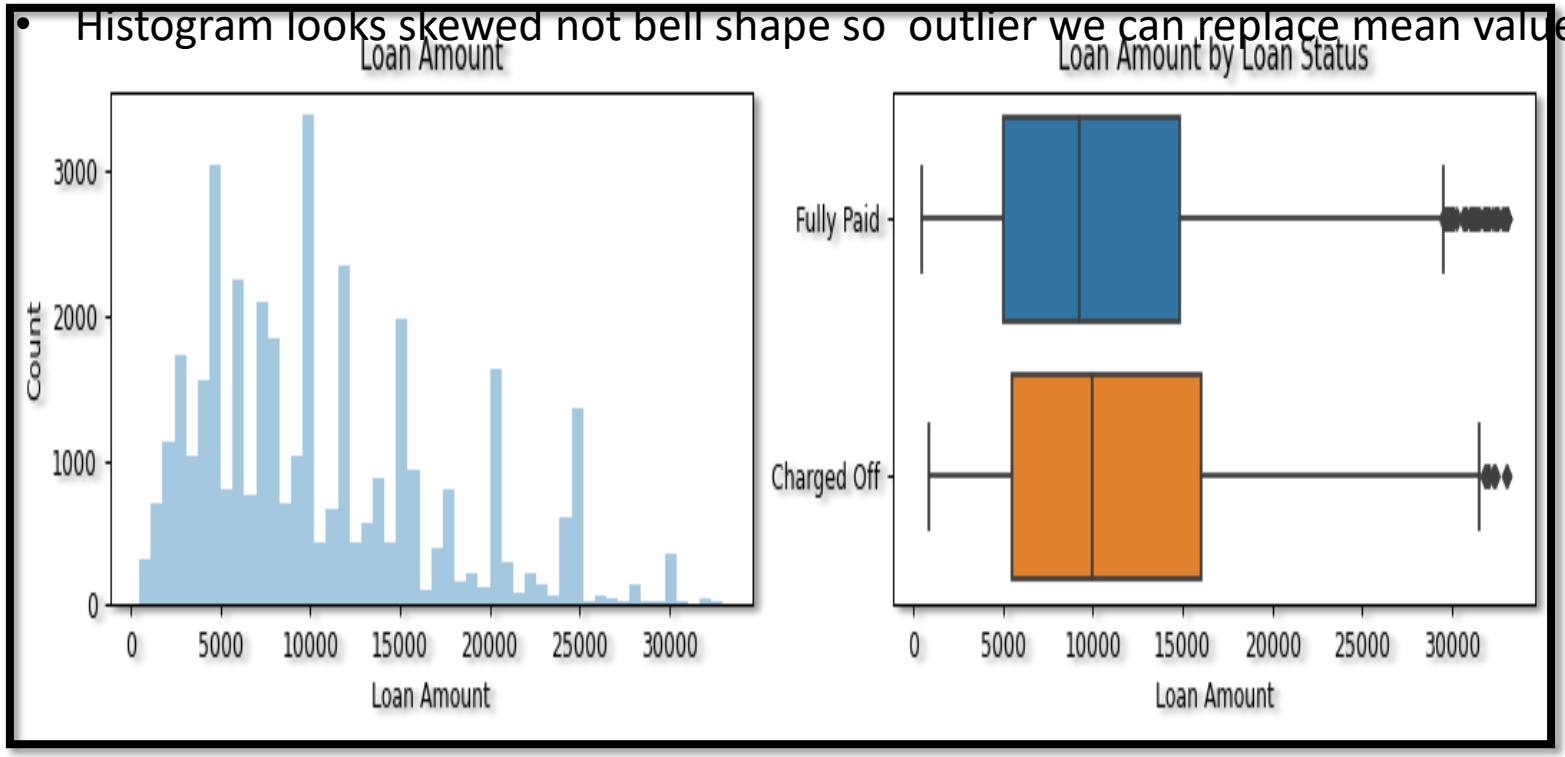
```
822
```

```
#drop the zip_code variable as there are too many categories present init and as it is not hlpful in our analysis
data.drop(labels='zip_code', axis=1, inplace=True)
```

```
data.groupby('addr_state')['loan_status'].value_counts(normalize=True).loc[:,'Charged Off'].sort_values()
```

Univariate Analysis Data Validation

- plotted the distribution plot and boxplot of loan_Amount variables hs looks skewed data and boxplot has outliers.
- Python Code: plot_var('loan_amnt', 'Loan Amount', continuous=True)
- Boxplot has too many outliers if skewness is appropriate and with in range then we can keep he outlier
- Histogram looks skewed not bell shape so outlier we can replace mean value




```
#check summary statistics by grouping loan_status wise
data.groupby('loan_status')['loan_amnt'].describe()
```

loan_status	count	mean	std	min	25%	50%	75%	max
Charged Off	5470.0	11448.642596	7200.001213	900.0	5500.0	10000.0	16000.0	33000.0
Fully Paid	32466.0	10507.655701	6621.319442	500.0	5000.0	9250.0	14800.0	33000.0

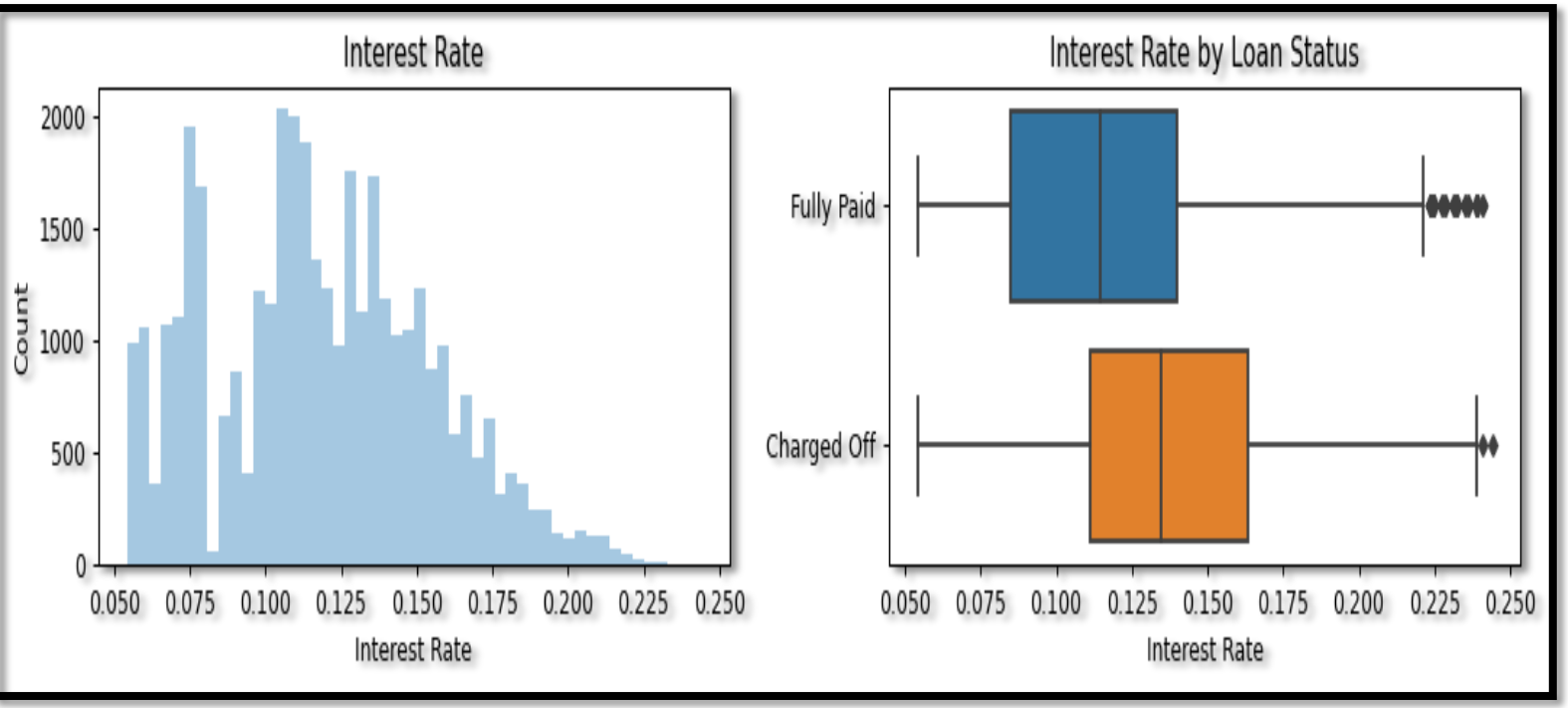
```
#check value counts of different categories of term variable
data['term'].value_counts(dropna=False)
```

```
 36 months    28926
 60 months     9010
Name: term, dtype: int64
```

```
data['term'].value_counts(normalize=True)
```

- Interest variables has looks skewed data and boxplot has outlier comparatively this better then previous loan amount visualization plot
- Below Boxplot and histogram is better comparatively previous slide but have some outliers if skewness is appropriate and with in range then we can keep he outlier
- Histogram looks skewed not bell shape so outlier we can replace mean value

plot_var('int_rate', 'Interest Rate', continuous=True)




```
print(data["int_rate"].skew())
print(data["int_rate"].kurt())
```

```
0.28310214992559934
-0.45703584433990674
```

```
data.groupby('loan_status')['int_rate'].describe()
```

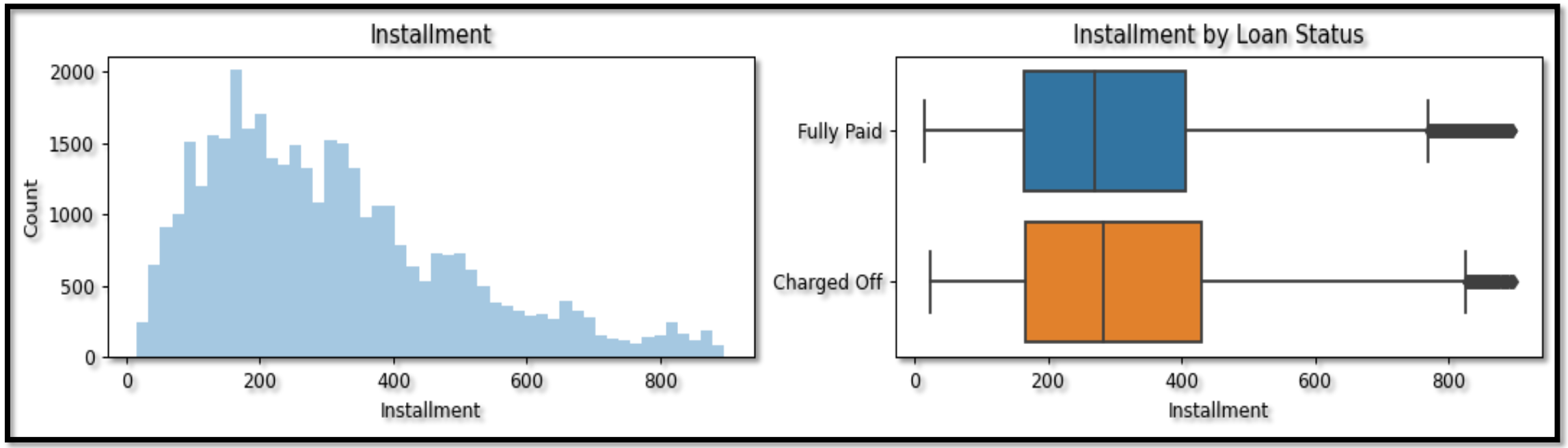
loan_status	count	mean	std	min	25%	50%	75%	max
Charged Off	5470.0	0.137160	0.036039	0.0542	0.1114	0.1349	0.1632	0.2440
Fully Paid	32466.0	0.115553	0.035662	0.0542	0.0849	0.1148	0.1398	0.2411

```
data['installment'].describe()
```

```
count    37936.000000
mean       312.681133
std        194.539773
min         15.690000
25%        164.550000
50%        273.760000
75%        412.800000
max       1231.450000
```

instalment data visualization

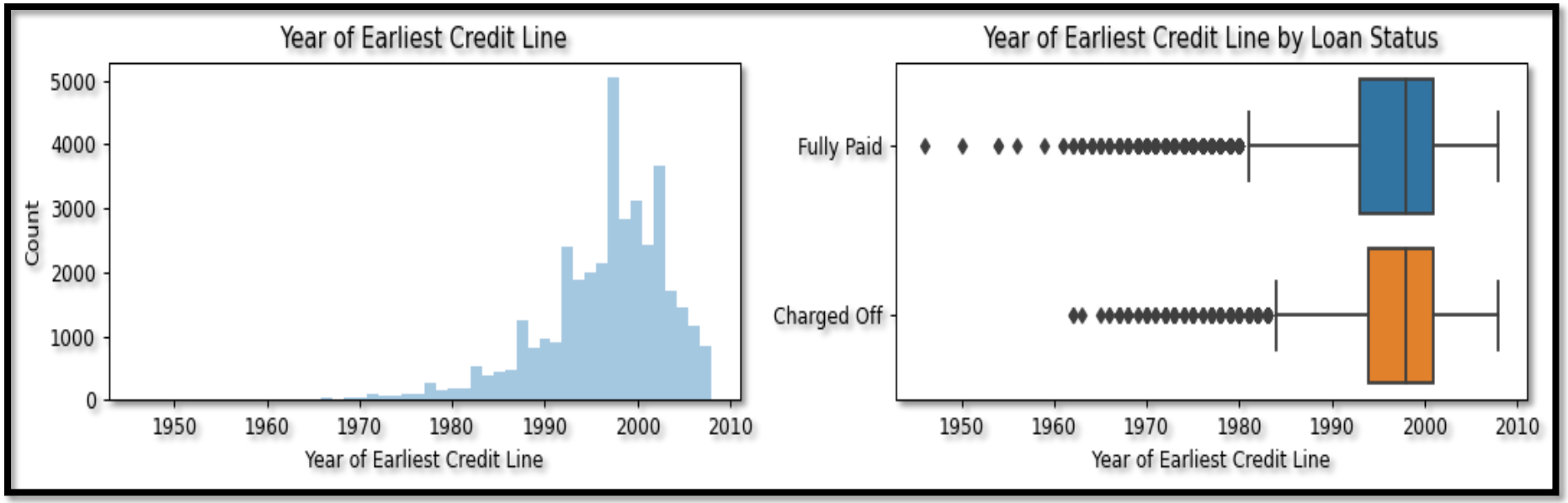
- previous loan amount visualization plot
- This Histogram Boxplot and is better then the previous both slides comparatively previous slide but have some outliers if skewness is appropriate and with in range then we can keep he outlier
- Histogram looks skewed not bell shape so outlier we can replace mean value

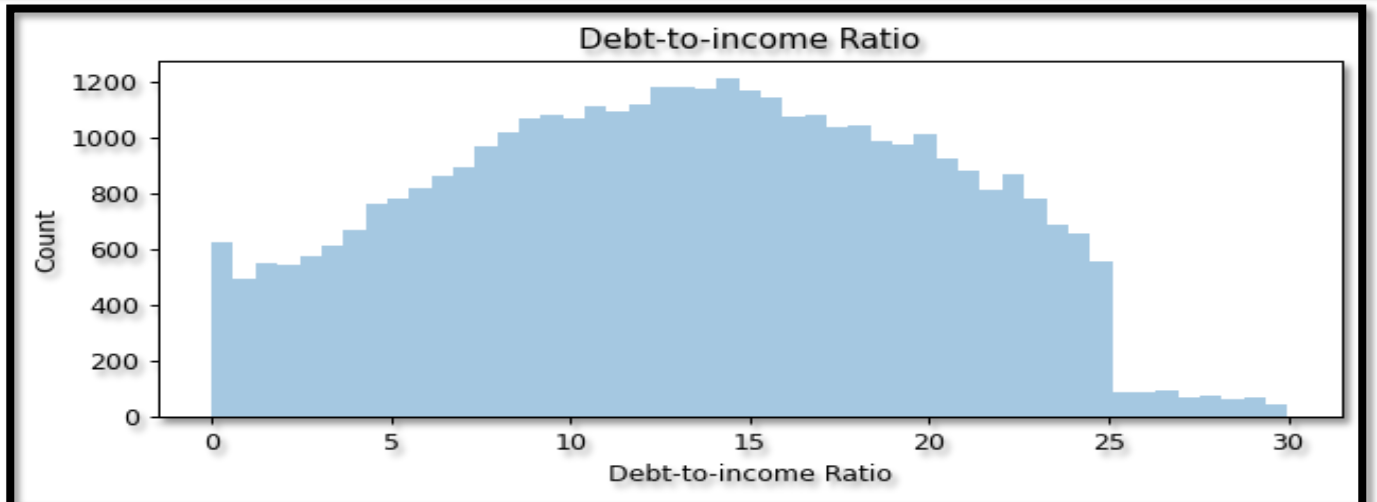


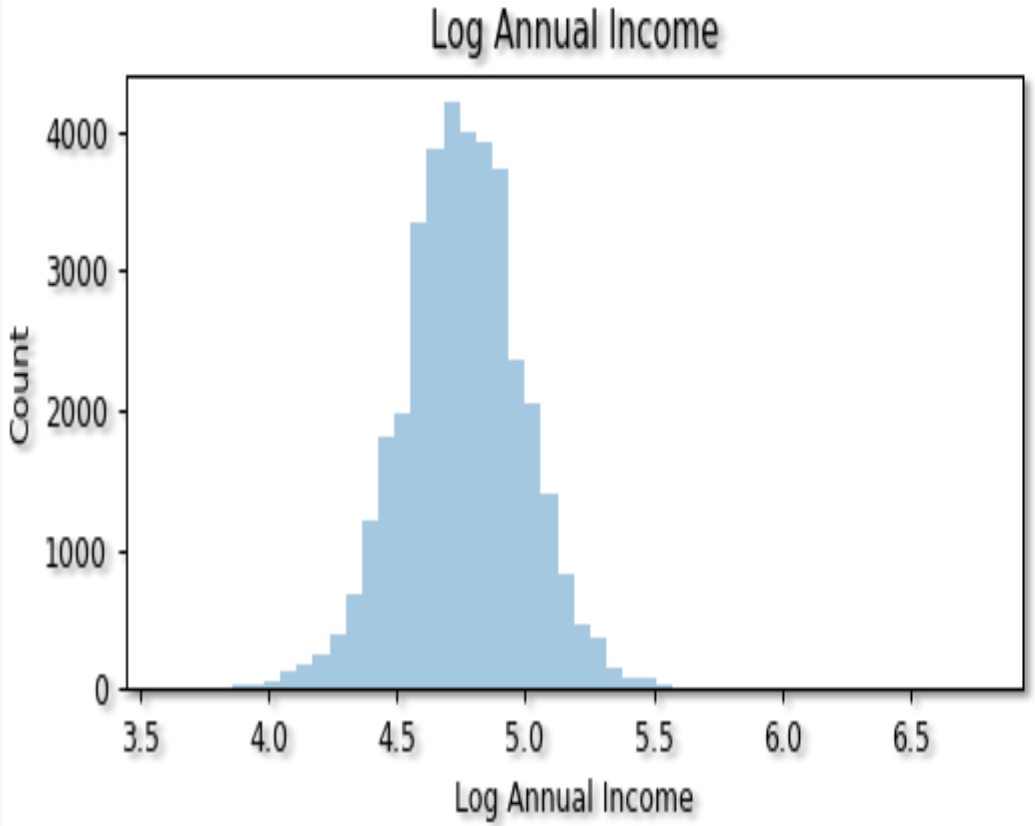

```
data.groupby('loan_status')['installment'].describe()
```

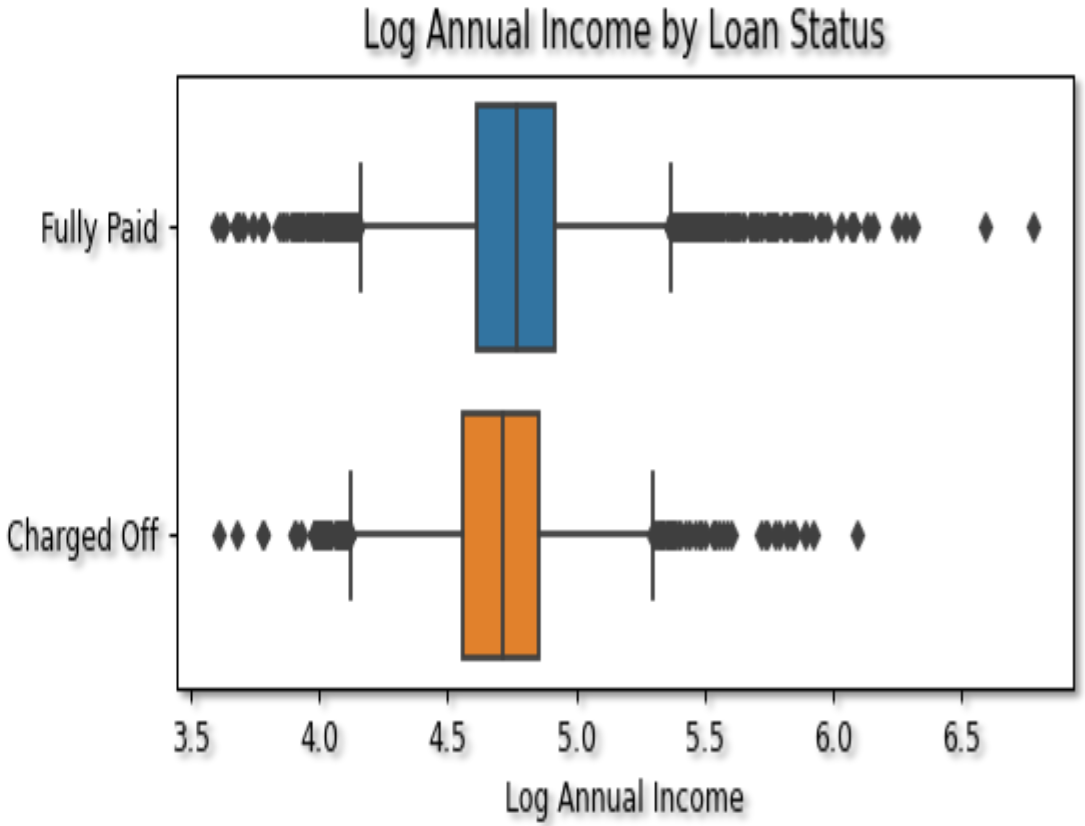
loan_status	count	mean	std	min	25%	50%	75%	max
Charged Off	5438.0	317.048814	192.313869	22.79	166.63	284.05	430.335	896.19
Fully Paid	32265.0	307.164734	187.169832	15.69	163.67	270.21	406.360	895.73

- Interest variables has looks skewed data and boxplot has outlier comparatively this better then previous loan amount visualization plot
- This Boxplot and histogram is right skewed comparatively previous slide but have some outliers if skewness is appropriate and with in range then we can keep he outlier
- Histogram looks skewed not bell shape so outlier we can replace mean value



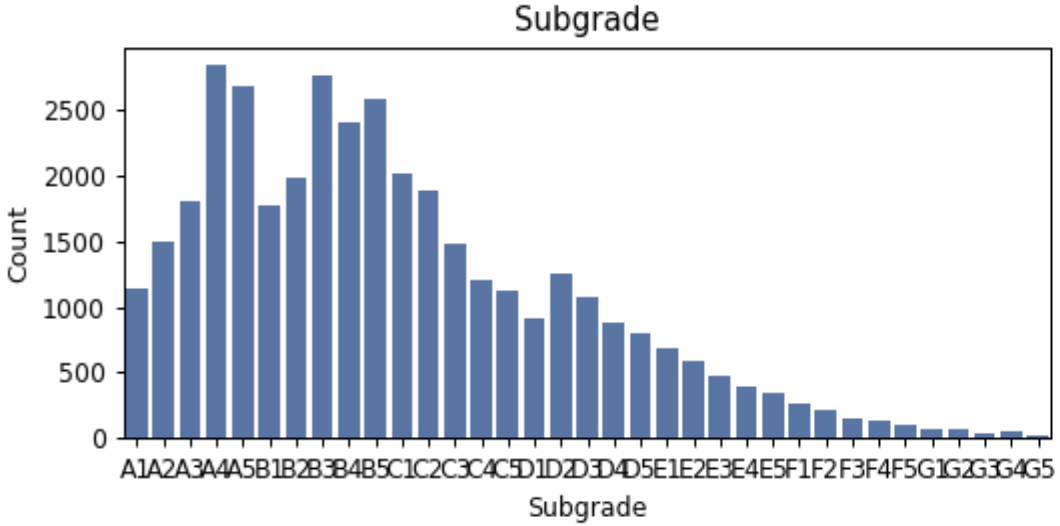


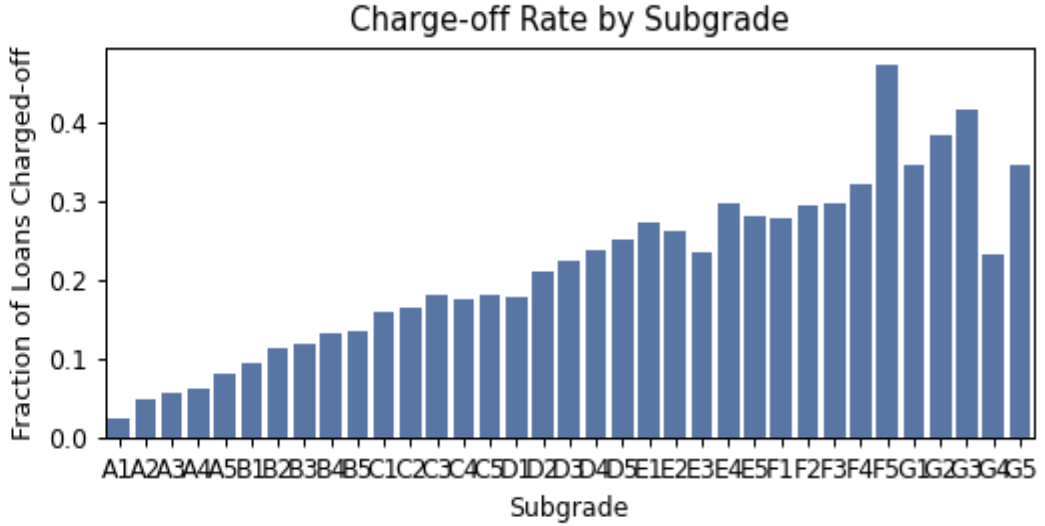


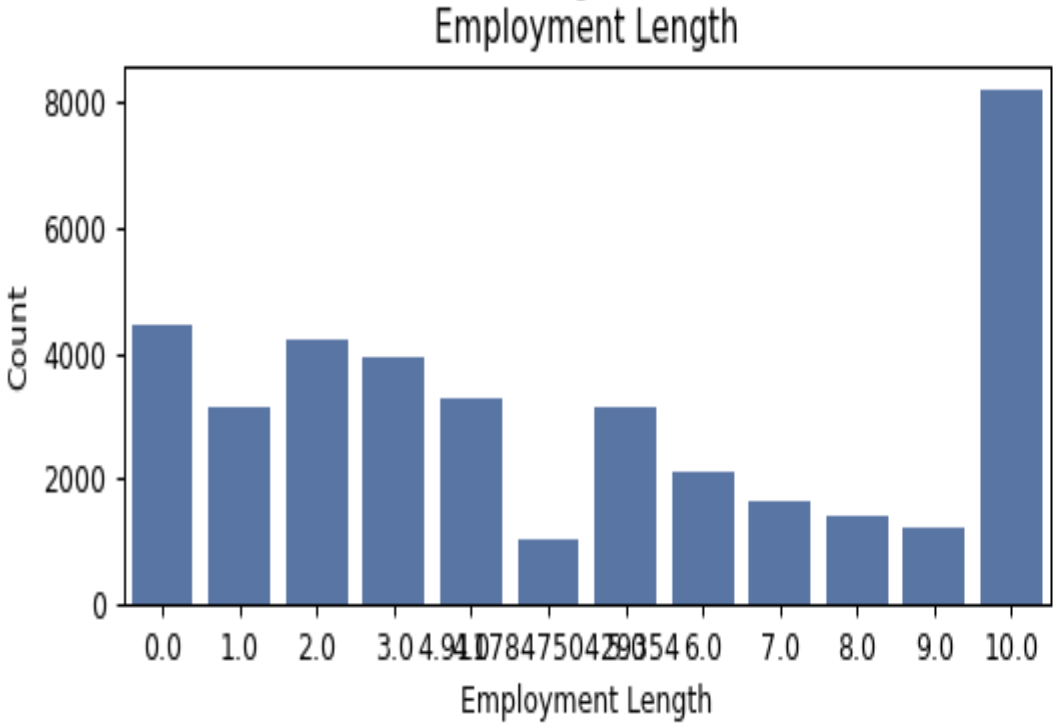


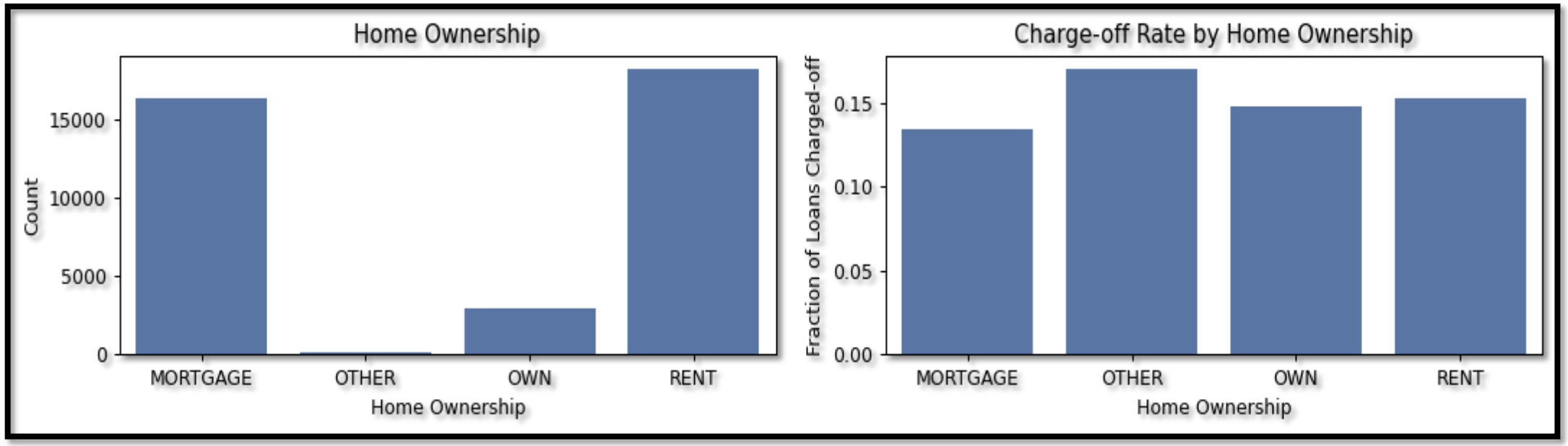
loan_status	count	mean	std	min	25%	50%	75%	max
Charged Off	5438.0	4.713915	0.239981	3.610767	4.561164	4.716012	4.857339	6.096910
Fully Paid	32265.0	4.764397	0.238985	3.602169	4.610671	4.770859	4.913819	6.778151

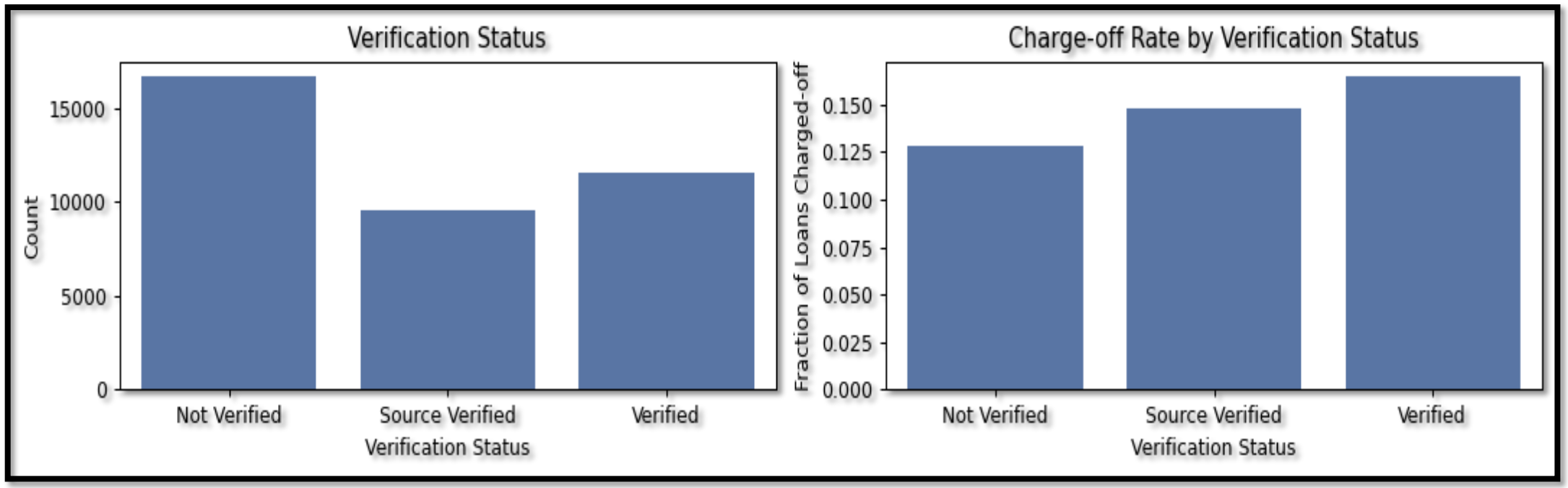
Visualization of Data



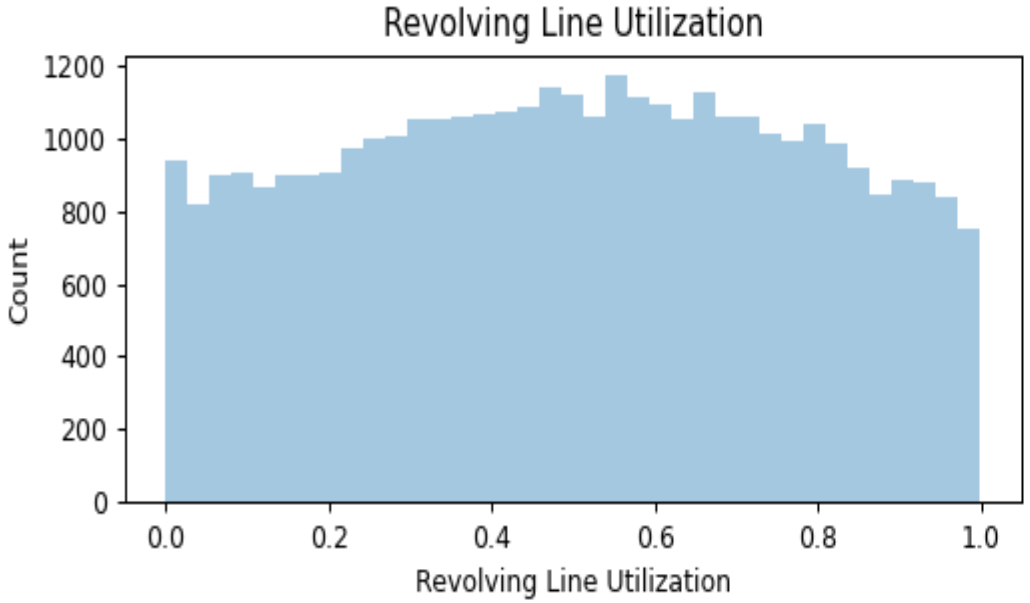


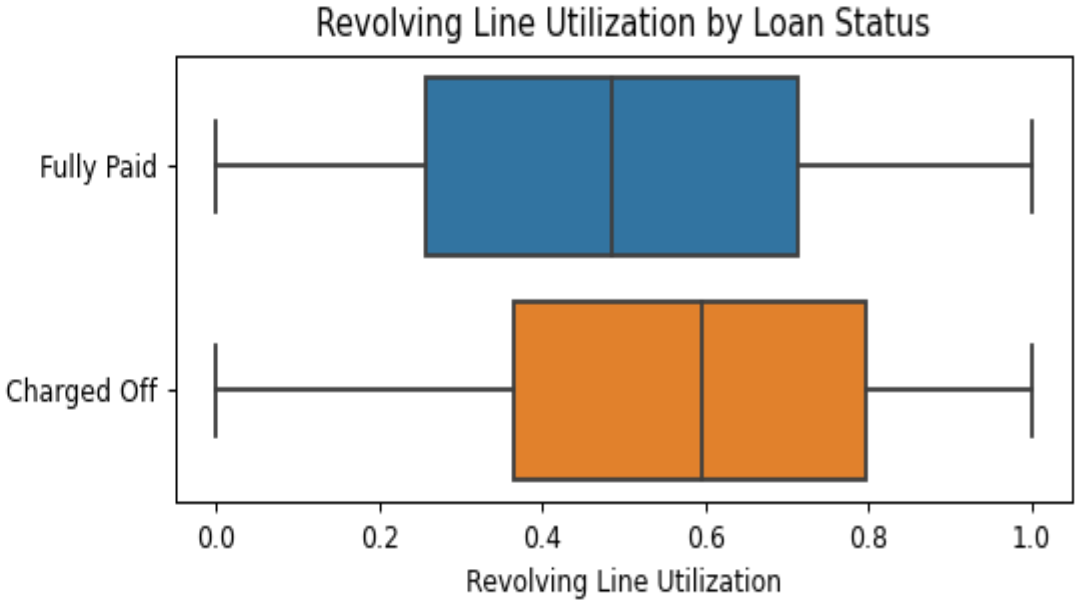


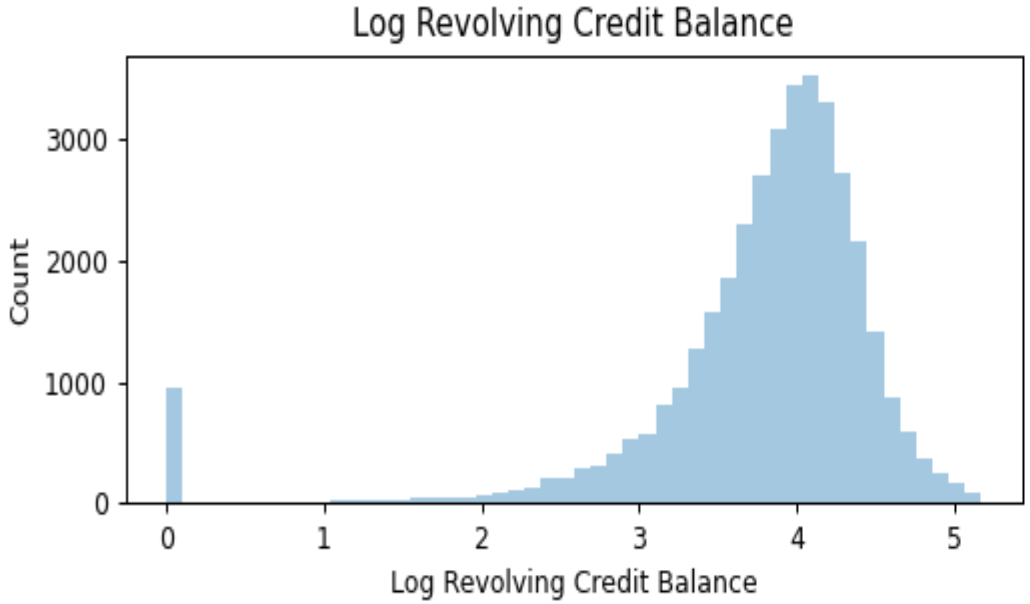

Home Ownership
Count
15000
10000
5000
0
MORTGAGE
OTHER
OWN
RENT
Home Ownership
Charge-off Rate by Home Ownership
Fraction of Loans Charged-off
0.15
0.10
0.05
0.00
MORTGAGE
OTHER
OWN
RENT
Home Ownership

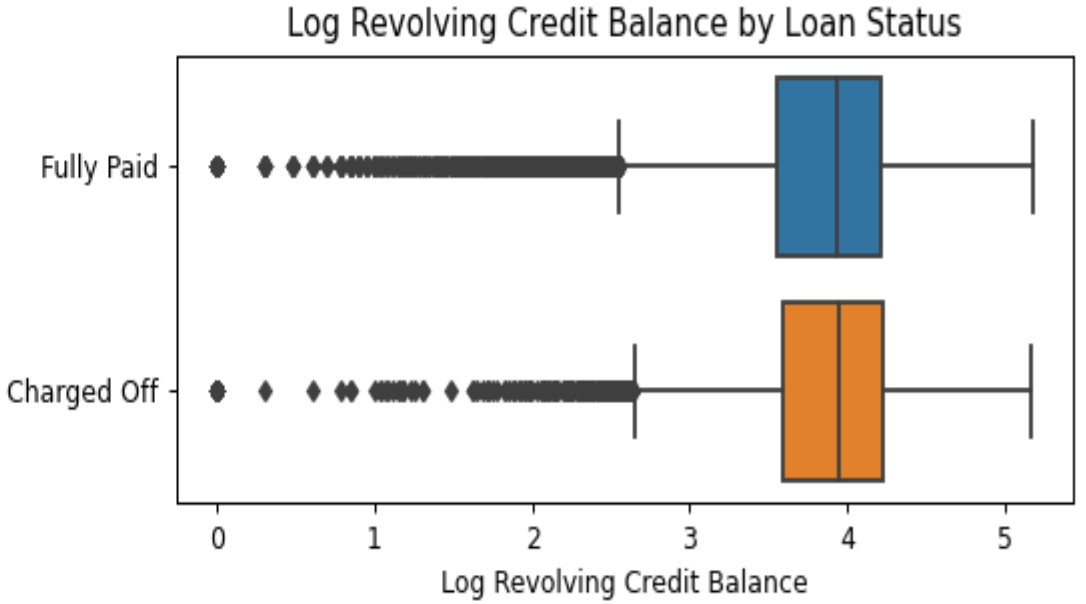

Verification Status
Count
15000
10000
5000
0
Not Verified
Source Verified
Verified
Verification Status
Charge-off Rate by Verification Status
Fraction of Loans Charged-off
0.150
0.125
0.100
0.075
0.050
0.025
0.000
Not Verified
Source Verified
Verified
Verification Status

After Outlier Treatment









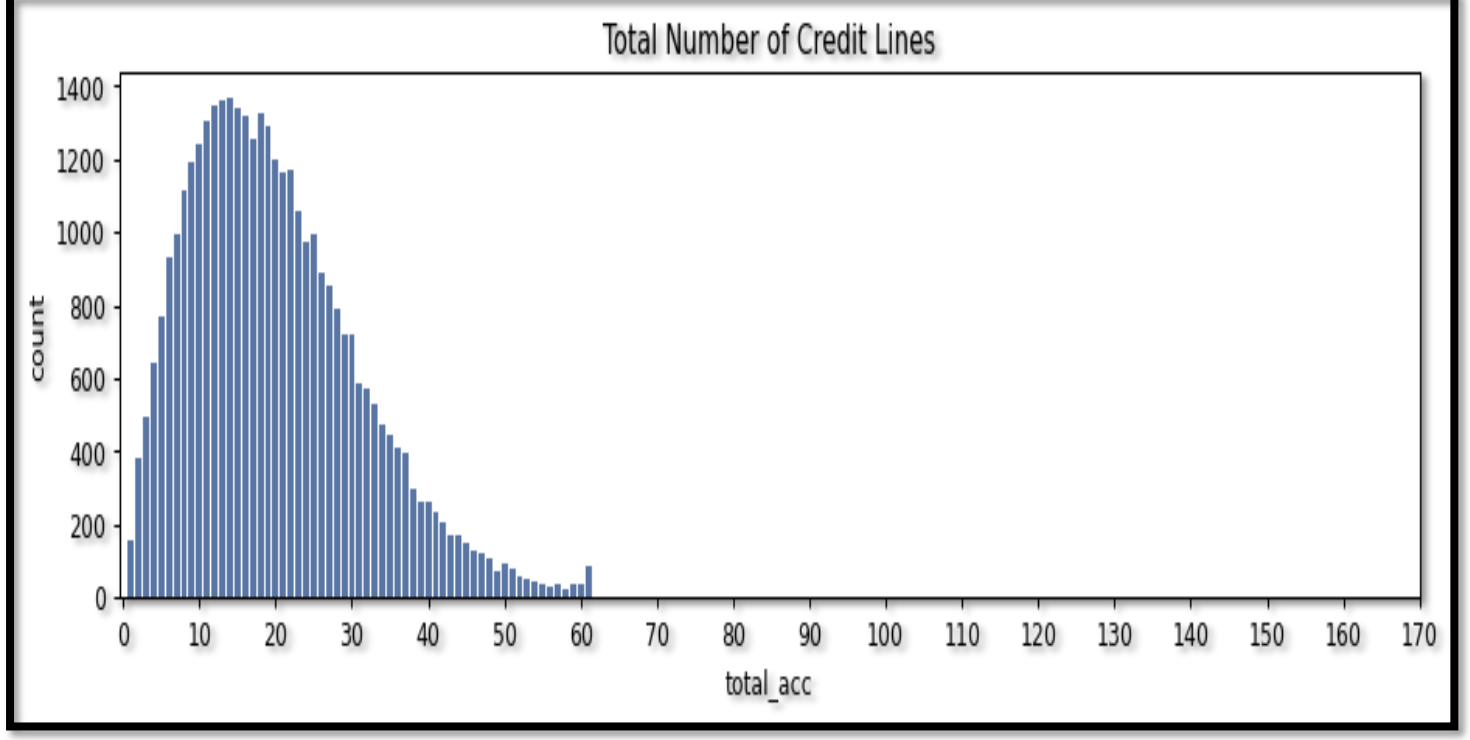
Dummy variable creation

```
Index(['loan_amnt', 'term', 'int_rate', 'installment', 'sub_grade',
       'emp_length', 'home_ownership', 'verification_status', 'issue_d',
       'purpose', 'addr_state', 'dti', 'earliest_cr_line', 'open_acc',
       'pub_rec', 'revol_util', 'total_acc', 'initial_list_status',
       'log_annual_inc', 'log_revol_bal', 'charged_off'],
      dtype='object')
```

```
#add dummies to above selected variables
data = pd.get_dummies(data, columns=['sub_grade', 'home_ownership', 'verification_status', 'purpose', 'addr_state', 'initial_lis
```

```
data.head()
```

	loan_amnt	term	int_rate	installment	emp_length	issue_d	dti	earliest_cr_line	open_acc	pub_rec	revol_util	total_acc	log_annual_inc	log_revol_bal
0	5000	36 months	0.1065	162.87	10.0	2011-12-01	27.65	1985	3	0	0.837	9	4.380229	4.135101
1	2500	60 months	0.1527	59.83	0.0	2011-12-01	1.00	1999	3	0	0.094	4	4.477136	3.227372
2	2400	36 months	0.1596	84.33	10.0	2011-12-01	8.72	2001	2	0	0.985	10	4.088242	3.470851
3	10000	36 months	0.1349	339.31	10.0	2011-12-01	20.00	1996	10	0	0.210	37	4.691974	3.748110
5	5000	36 months	0.0790	156.46	3.0	2011-12-01	11.20	2004	9	0	0.283	12	4.556315	3.901131



plot the count plot of open_acc
plt.figure(figsize=(10,3), dpi=90)
sns.countplot(data['open_acc'],
order=sorted(data['open_acc'].unique()),
color='#5975A4', saturation=1)
_, _ = plt.xticks(np.arange(0, 90, 5),
np.arange(0, 90, 5))
plt.title('Number of Open Credit Lines')

In [91]:
```
#finding summary statistics of total_acc after grouping the data by loan_status
data.groupby('loan_status')['total_acc'].describe()
```

Out[91]:

loan_status	count	mean	std	min	25%	50%	75%	max
Charged Off	5267.0	21.453389	11.379642	2.0	13.0	20.0	28.0	74.0
Fully Paid	31374.0	22.119207	11.385226	2.0	14.0	20.0	29.0	90.0

In [92]:
```
#plot distplot of 'initial_list_status'
plot_var('initial_list_status', 'Initial List Status', continuous=False)
```

Correlation Plot

	loan_amnt	int_rate	installment	emp_length	dti	earliest_cr_line	open_acc	pub_rec	revol_util	total_acc	log_annual_inc	log_revol_b
loan_amnt	1.000000	-0.264646	0.995587	0.418850	0.598394	-0.226196	0.811551	NaN	-0.343395	0.940489	0.744423	0.45919
int_rate	-0.264646	1.000000	-0.238917	0.111001	-0.436146	0.085413	-0.511534	NaN	0.121659	-0.003605	-0.423026	-0.79027
installment	0.995587	-0.238917	1.000000	0.495511	0.625268	-0.231539	0.786859	NaN	-0.263385	0.954174	0.679790	0.47335
emp_length	0.418850	0.111001	0.495511	1.000000	0.733768	-0.493907	-0.016614	NaN	0.688050	0.487526	-0.278515	0.47885
dti	0.598394	-0.436146	0.625268	0.733768	1.000000	-0.782770	0.217084	NaN	0.371801	0.423650	0.186068	0.88406
earliest_cr_line	-0.226196	0.085413	-0.231539	-0.493907	-0.782770	1.000000	0.280193	NaN	-0.353311	-0.034250	-0.008031	-0.54896
open_acc	0.811551	-0.511534	0.786859	-0.016614	0.217084	0.280193	1.000000	NaN	-0.593301	0.755228	0.824488	0.33665
pub_rec	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
revol_util	-0.343395	0.121659	-0.263385	0.688050	0.371801	-0.353311	-0.593301	NaN	1.000000	-0.277036	-0.836469	0.29650
total_acc	0.940489	-0.003605	0.954174	0.487526	0.423650	-0.034250	0.755228	NaN	-0.277036	1.000000	0.586335	0.22597
log_annual_inc	0.744423	-0.423026	0.679790	-0.278515	0.186068	-0.008031	0.824488	NaN	-0.836469	0.586335	1.000000	0.22443
log_revol_bal	0.459195	-0.790279	0.473352	0.478857	0.884061	-0.548964	0.336650	NaN	0.296500	0.225979	0.224434	1.00000
charged_off	-0.449748	0.435572	-0.514404	-0.772683	-0.688193	0.152854	-0.354787	NaN	-0.540633	-0.448143	0.094554	-0.73598

THANK YOU